Szerkezeti rekurziók élcímkézett gráfok fölött

Doktori értekezés tézisek

Kósa Balázs

Témavezető: Benczúr András D.Sc.

Eötvös Loránd Tudományegyetem Informatikai Kar
Információs Rendszerek Tanszék

Informatikai Doktori Iskola
Benczúr András D.Sc.

Információs Rendszerek Doktori Program
Benczúr András D.Sc.

Budapest, 2013

1. Bevezetés

A disszertációban egy gráftranszformátort vezetünk be, melyre a rövidség kedvéért *szerkezeti rekurzióként* hivatkozunk, majd ennek vizsgáljuk meg néhány tulajdonságát. A szerkezeti rekurzió gyakorlati fontosságát a féligstrukturált és XML adatbázisok területén való alkalmazhatósága adja, ahol az adatokat címkézett gráfokkal és fákkal reprezentálják. Általánosságban a szerkezeti rekurziókat olyan (matematikai) objektumokon alkalmazzák, melyeket "konstruktorok" segítségével lehet felépíteni. Az ítéletlogikában ilyen konstruktorok például a $\neg, \wedge, \vee$ logikai összekötők. Az adatbázisok világában a szerkezeti rekurziók már a 90-es évek elején megjelentek egy olyan lekérdezőnyelv alapjaként, amely meghaladja relációs algebra korlátait és egy általános célú programozási nyelvbe beágyazott relációs lekérdezési nyelv kifejezőerejének lehetőségeihez közelít [7]. A szerkezeti rekurziók itteni változata halmazok fölött lett definiálva. Az elkövetkező években a szerkezeti rekurziókat halmaz-, multihalmaz és listaalapú (beágyazott) kollekciók lekérdezésére alkalmazták [10][13][12]. Ezekben a munkákban azonban a szerkezeti rekurzióknak egy általánosabb formáját használták, mint ami a disszertációban szerepel. A mi megközelítésünket közvetlenül a féligstrukturált adatok kezelésére bevezetett UnQL nyelv esetében adott definíció inspirálta [9]. Itt a szerkezeti rekurzók élcímkézett gráfokat dolgoznak fel, és az UnQL mögötti UnCal-nak nevezett algebra fő műveletét alkotják.

Korábbi eredmények.. Mindazonáltal a disszertációban szereplő definíció lépésről-lépésre alalult a különböző XML lekérdezőnyelvek szimulációjakor fellépő igények hatására. Emiatt hasznosnak bizonyulhat, ha végigvesszük a folyamat főbb állomásait.

Első cikkünkben [1] az élcímkézett gráfokat feldolgozó szerkezeti rekurziók szemantikáját újfajta módon definiáltuk. A [9] cikkben alkalmazott és az imént említett szemantika ekvivalenciája bebizonyítható. Emellett megmutattuk, miként működhetnek együtt a szerkezeti rekurziók az adatok szerkezetére vonatkozó megszorításokkal. Ezeket a megszorításokat sémagráfok segítségével fejezik ki [8].

Második cikkünkben [2] feltételek megadásának lehetőségével bővítettük

a szerkezeti rekurziót, melyekben *isempty* és *not-isempty* logikai függvényeket és egyenlőséget használhatunk. Kis csúsztatással az *isempty*, *not-isempty* függvényekben egy másik szerkezeti rekurzió eredményének ürességét el lenőrizhetjük, melyet az éppen feldolgozott él alatti részgráfon hívunk meg. Az ily módon kiterjesztett szerkezeti rekurziók segítségével sikeresen szimuláltuk az XPath 1.0 lekérdezőnyelv egy magját. Az [1]-ben megjelenő tárgyalásmódhoz hasonlóan leírtuk, miként lehet a DTD-k (Document Type Definitions (DTD) [6]) és EDTD-k (Extended Document Type Definitions (EDTD) [14]) adta megszorításokat is átvezetni a szimuláló szerkezeti rekurziókba. A szimuláció egyben egy $O(|D||Q|)$ időben működő, tehát hatékony megvalósítást is implikál, ahol $|D|$ az adat, míg $|Q|$ a lekérdezés méretét jelöli. Ez megegyezik a Gottlob és munkatársai által javasolt algoritmus sebességével [11].

[5]-ban az iménti szimulációt terjesztettük ki az XSLT 1.0 egy magjára. A változók jelenléte, melyekhez XPath kifejezések segítségével rendelhetünk értéket, meglehetősen bonyolulttá tette a részleteket. Ebben az esetben a szimuláció nyújtotta megvalósítás $O(|D|^2|Q|^2)$ időben működik.

A következő cikkünkben [3] a statikus analízishez tartozó szokásos kielégíthetőségi és tartalmazási kérdéseket vizsgáltuk. A szerkezeti rekurzióknak azonban csupán egy szűkített változatát elemeztük, ahol a feltételekben egyetlen *not-isempty* logikai függvény használata volt engedélyezett. Más szavakkal sem az *isempty* feltétel, sem egyenlőségek megadása nem volt lehetséges. A másik oldalról viszont két esetet különböztettünk meg az else ágak alkalmazhatósága alapján. Kiderült, hogy else ágak nélkül a kielégíthetőségi kérdés négyzetes időben megoldható, míg a tartalmazás coNP-nehéz. Az else ágak esetében pedig megmutattuk, hogy a két probléma visszavezethető egymásra és mind a kettő PSPACE-nehéz.

Végül [4]-ban az XPath kifejezéseket *nevesített kifejezések* megadásának lehetőségével bővítettük ki, melyek segítségével – amint a név is mutatja – nevet rendelhetünk egy XPath kifejezéshez. Egy nevesített kifejezésben önmaga vagy más nevesített kifejezések is szerepelhetnek, melynek köszönhetően a `descendant`, `ancestor` stb. tengelyek adta rekurzivitástól eltérő rekurzivitás is megjelenik. Az ötletet az XQuery függvény definiálási lehetősége inspirálta, melyeknek törzsében más felhasználó által megadott függvényeket is meghívhatunk. A nevesített kifejezéseket az iménti függvények

egyszerűsített változatának tekinthetők az XPath kontextusában.

A disszertációban a szerkezeti rekurziók matematikai elemzése lett lényegesen továbbfinomítva. Az itt megjelenő és a fentiekben említett összes cikk [1][2][5][3][4] eredménye a sajátom, mindazonáltal mélyen hálás vagyok társszerző kollégáim Benczúr András D.Sc. és Kiss Attila C.Sc. hasznos meglátásaiért és tanácsaiért.

2. Fő eredmények

A strukturális rekurzió egy négyes, $f = (F, \Sigma, F_I, \Gamma)$, ahol $F = \{f_1, \ldots, f_n\}$ a *szerkezeti függvények*, $F_I = \{f_{i_1}, \ldots, f_{i_k}\}$ a *kezdő* szerkezeti függvények, Σ az f által feldolgozandó adatgráfokban alkalmazható élcímkék, Γ pedig a *transzformációs szabályok* halmaza. Adatfák esetében a szerkezeti rekurziók felülről lefelé haladva dolgozzák fel a bemenetüket, adatgráfoknál a működés hasonló, emellett minden kört csupán egyszer járunk be. Egy transzformációs szabálynál azt adjuk meg, miként viselkedjen egy szerkezeti függvény, amikor például egy adott címkéjű élhez ér. Ilyen esetben a szerkezeti függvény egy erdőt konstruálhat, melynek leveleihez a szóban forgó él alatti részgráfra meghívott szerkezeti függvények eredményeit kapcsolhatja. A feltételek *if-then-else* utasítások formájában adhatók meg.

A disszertációban a bevezetés után a 2. fejezetben az alapfogalmakat ismertetjük, míg a 3. fejezetben a szerkezeti rekurziók szemantikáját adjuk meg. A definíció távolról sem nyilvánvaló, mivel a feltételek kiértékelési módjának meghatározása a körök jelenléte miatt kifinomult tervezést igényel. A 4. fejezetben a szerkezeti rekurziók öt osztályát különböztetjük meg egymástól, majd összehasonlítjuk ezek kifejezőerejét. $SR(n.i., i., el)$ jelöli azt az osztályt, ahol mind a *not-isempty* (n.i.), az *isempty* (i) feltételek, mind az else ágak használata megengedett a transzformációs szabályokhoz tartozó feltételekben. Hasonlóképpen $SR(n.i., i.)$ jelöli azt az osztályt, ahol *not-isempty* és *isempty* feltételek is megjelenhetnek, else ágak azonban már nem. Emellett az $SR(n.i.)$ és $SR()$ jelölések szintén előfordulnak, melyeknek értelme az iménti magyarázatok fényében már világos. Egy $SR()$-beli szerkezeti rekurzió *determinisztikus* továbbá, ha csupán egyetlen kezdő szerkezeti függvény tartozik hozzá, a transzformációs szabályaiban pedig legfeljebb egy szerkezeti

függvény kerül meghívásra. A determinisztikus szerkezeti rekurziók osztályát *Det* jelöli.

Egy f szerkezeti rekurzió esetében jelölje $\mathcal{L}(f)$ azon adatgráfok osztályát, melyekre f nem üres eredményt ad. Szerkezeti rekurziók két $\mathcal{SR}_1, \mathcal{SR}_2$ osztályára $\mathcal{SR}_1$ kifejező ereje *erősebb*, ha minden f^2 $\mathcal{SR}_2$-beli szerkezeti rekurzióhoz létezik f^1 szerkezeti rekurzió $\mathcal{SR}_1$-ben, amire $\mathcal{L}(f^1)$ megegyezik $\mathcal{L}(f^2)$-vel. $\mathcal{SR}_1$ *szigorúan erősebb*, ha erősebb, mint $\mathcal{SR}_2$, ráadásul létezik olyan $\mathcal{SR}_1$-beli g^1 szerkezeti rekurzió, amihez nem adható meg egyetlen olyan $\mathcal{SR}_2$-beli g^2 szerkezeti rekurzió sem, amire $\mathcal{L}(g^2)$ megegyezne $\mathcal{L}(g^1)$-gyel. Végül $\mathcal{SR}_1$ *azonos* lekérdező erejű $\mathcal{SR}_2$-vel, ha $\mathcal{SR}_1$ erősebb, mint $\mathcal{SR}_2$ és megfordítva.

A disszertációban a 4.16., 4.19., 4.20. állítások valamint az 5.11. következmény az iménti viszonyokat tisztázza a szerkezeti rekurzió osztályok között.

Tézis 1. *(i) A determinisztikus szerkezeti rekurziók osztályának azonos a kifejező ereje a feltétel nélküli szerkezeti rekurziók osztályáéval* ($SR()$).

(ii) A not-isempty *feltételekkel rendelkező szerkezeti rekurziók osztálya* ($SR(n.i.)$) *szigorúan erősebb, mint* $SR()$.

(iii) A not-isempty *és* isempty *feltételekkel is rendelkező szerkezeti rekurziók osztálya* ($SR(n.i., i.)$) *szigorúan erősebb, mint* $SR(n.i.)$.

(iv) A not-isempty *és* isempty *feltételekkel valamint else ágakkal is rendelkező szerkezeti rekurziók osztálya* ($SR(n.i., i., el)$) *azonos kifejező erejű, mint* $SR(n.i., i.)$.

Ezt követően a 4.4. fejezetben egy adott szerkezeti rekurzióra a bemeneti adatgráfokat adatfákkal "szimuláljuk", ahol az adatgráf feldolgozása hasonló módon történik, mint a szimulációjáé. (A hasonlóság mikéntjét egzakt definíció taglalja.) Az eredményeket a 4.45. állítás foglalja össze.

Tézis 2. *Adott f szerkezeti rekurzió esetében, ha létezik olyan adatgráf, amire f nem üres ereményt ad, akkor létezik olyan adatfa is, amire f nem üres eredménnyel tér vissza.*

Ezek után az 5. fejezetben a szokásos műveleteket: komplementerképzés ($\tilde{}$), unió ($\sqcup$), metszet ($\sqcap$) vezetjük be. Egy f szerkezeti rekurzióra jelölje $\mathcal{L}_{root}(f)$ azon *gyökéréles* adatgráfok osztályát, melyeken f nem üres eredménnyel tér vissza. Egy adatgráf gyökéréles, ha a gyökerének egyetlen kimenő éle van. Az 5.8. és 5.14. állítások a műveletek és azon gyökéréles adatgráfok osztályainak a viszonyát tisztázzák, melyeknek elemein az operandusok egyike nem üres eredményt ad.

Tézis 3. *Tetszőleges f és g szerkezeti rekurziókra:*

(i) $\widetilde{\mathcal{L}_{root}(f)}$ *ugyanaz, mint* $\mathcal{L}_{root}(\tilde{f})$.

(ii) $\mathcal{L}_{root}(f \sqcap g)$ *megegyezik* $\mathcal{L}_{root}(f) \cap \mathcal{L}_{root}(g)$*-vel.*

(iii) $\mathcal{L}(f \sqcup g)$ *ugyanaz, mint* $\mathcal{L}(f) \cup \mathcal{L}(g)$.

Itt $\mathcal{L}(f)$ ($\mathcal{L}_{root}(f)$) *azon adatgráfok osztályát jelöli, melyre f nem üres eredménnyel tér vissza.*

Emellett az 5.17. állításban kiderül, hogy a De Morgan azonosságok szintén teljesülnek a szerkezeti rekurziók iménti műveletei között.

Hogy mélyebben megérthessük a szerkezeti rekurziók működését a 6. fejezetben két jól ismert automata típussal – nemdeterminisztikus, véges állapotú sztring automaták és alternáló faautomaták – hasonlítjuk össze őket. Az elfogadást a nem üres végeremény reprezentálja és megfordítva. Egy Σ-beli w szóra jelölje pa_w azt az utat, melynek élei w szimbólumaival címkézettek a w-beli sorrendnek megfelelően. Megfordítva egy pa útra jelölje w_{pa} azt a szót, amiben a szimbólumok pa élcímkéi által adott sorrendnek megfelelően követik egymást. A 6.2. és 6.4. állítások a nemdeterminisztikus véges állapotú automaták és a szerkezeti rekurziók közötti viszonyt tisztázzák.

Tézis 4. *(i) Tetszőleges feltétel nélküli f szerkezeti rekurzióra, létezik A_f nemdeterminisztikus, véges állapotú automata, amire f akkor és csak akkor ad nem üres eredményt a pa útra, ha A_f elfogadja w_{pa}-t.*

(ii) Létezik olyan nemdeterminisztikus, véges állapotú automata A, amelyhez nem létezik olyan f feltétel nélküli szerkezeti függvény, ami hasonló módon szimulálná A-t, amiként a szerkezeti rekurziókat szimulálták a nemdeterminisztikus, véges állapotú automaták az imént.

Hogy megfelelően mérlegelhessük a tézis jelentőségét, érdemes megjegyezni, hogy ha egy feltételek nélküli szerkezeti rekurzió nem üres eredményt ad egy I adatgráfra, akkor I biztosan tartalmaz olyan gyökeréből induló utat, melyre f szintén nem üres eredménnyel tér vissza. Másodsorban, a tézis második pontja a nem üres eredménnyel való visszatérés, mint tulajdonság monotonitásából következik. Tudniillik, ha egy feltételek nélküli f szerkezeti rekurzió nem üres eredményt ad I adatgráfra, akkor f minden olyan adatgráfra, amelynek I részgráfja oly módon, hogy a gyökerek egybeesnek, szintén nem üres eredménnyel tér vissza. A másik oldalról viszont egy nemdeterminisztikus, véges állapotú automata nem feltétlenül fogadja el $w.a$-t, ha w-t el is fogadja, ahol $w.a$ azt a szót jelöli, melyet w-ből kapunk az a szimbólum hozzáadásával.

Az alternáló faautomatákkal való összehasonlítás jóval bonyolultabb, mivel ezek rangolt[1], rendezett[2], pontcímkézett fákon dolgoznak, míg a szerkezeti rekurziók rangolatlan, rendezés nélküli, élcímkézett gráfokon. A szerkezeti rekurziók alternáló faautomatákkal történő reprezentációjához egy f szerkezeti rekurzióra nem üres eredményt adó I adatgráf köreit fák egy halmazával reprezentáljuk, melyekre f szintén nem üres eredménnyel tér vissza. Ezek a fák minden lehetséges módját leírják az I-n történő konstrukcióknak az f általi feldolgozás során. A bonyodalmak azonban még nem oldódnak meg az iménti helyettesítéssel, mert egy rangolatlan fát nem lehet rangolttá átírni úgy, hogy ez a transzformáció az összes szerkezeti rekurzió esetén megfelelően működne. Emiatt az iménti átalakítást minden egyes szerkezeti rekurzióra külön kell megadni. Jelölje $\mathcal{I}_{f,I}$ az ily módon transzformált fák halmazát.

A szimuláció ellentétes irányánál, egy rangolt, rendezett, pontcímkézett t fa esetén jelölje $Tr(t)$ a rangolatlan, élcímkézett fává történő transzformálás eredményét, melyben t pontjainak sorrendje is reprezentálódik. Az alternáló faautomata és a szimuláló szerkezeti rekurzió közötti viszony leírásához a Tr transzformáció mellett ennek "inverzére", $ReTr$, is szükség lesz. Egy kis csúsztatással: egy rangolatlan, élcímkézett t fa esetén $ReTr(t)$ azon rangolt, rendezett, pontcímkézett t' fák halmazával egyenlő, melyekre $Tr(t')$ t-vel

[1]Egy adott címkéjű pont meghatározott számú kimenő éllel rendelkezik a címkének megfelelően.

[2]A rendezés a pontok között definiált.

azonos. A következő tézis a 6.10. és 6.16. tételek eredményeit összegzi.

Tézis 5. *(i) Tetszőleges f szerkezeti rekurzióhoz létezik A_f alternáló faautomata úgy, hogy f nem üres eredményt ad I adatgráfra akkor és csak akkor, ha A_f $\mathcal{I}_{f,I}$ összes elemét elfogadja.*

(ii) Legyen A alternáló faautomata. Jelölje f_A az A-t szimuláló szerkezeti rekurziót. Ekkor

a) tetszőleges rangolt, rendezett, pontcímkézett t fára, A akkor és csak akkor fogadja el t-t, ha $f_A(Tr(t))$ nem üres.

b) Egy gyökéréles, rangolatlan, élcímkézett t fa esetén $f_A(t)$ akkor és csak akkor nem üres, ha A elfogadja $ReTr(t)$ összes elemét.

Végül a 7. és 8. fejezetekben a szokásos statikus analízisbeli kérdéseket vizsgáljuk. A kielégíthetőségi probléma esetén azt kérdezzük, hogy adott szerkezeti rekurzióhoz létezik-e olyan adatgráf, melyre a szerkezeti rekurzió nem üres eredménnyel tér vissza. A tartalmazási problémánál f és g szerkezeti rekurziókra azt ellenőrizzük, hogy minden adatgráfra, melyre g nem üres eredményt ad, f szintén nem üres eredménnyel tér-e vissza. Emellett a tartalmazás egy szigorúbb változatát is górcső alá vesszük. f és g szerkezeti rekurziókra f *szigorúan tartalmazza* g-t, ha minden I adatgráfra és annak minden e élére, ha g konstruál e-n I feldolgozása során, akkor f szintén konstruál e-n. Ez a különbségtétel az XPath kifejezések esetében alkalmazott hasonló megkülönböztetésen alapul [15]. Mindazonáltal megmutatható, hogy a kétféle típusú tartalmazási feladat polinom időben visszavezethető egymásra (4.8. és 4.12. állítások).

Tézis 6. *A a szerkezeti rekurziók osztályaira vonatkozó kielégíthetőségi és tartalmazási kérdésekre vonatkozó bonyolultságelméleti eredmények a 1. ábrán találhatók. Az $SR(n.i.)$ osztály tartalmazási és az $SR(n.i., i., \leq k)$ osztály kielégíthetőségi és tartalmazási kérdésein kívül minden a táblázatban szereplő probléma teljes a megfelelő bonyolultsági osztályra nézve (7.9. következmény, 7.12., 7.25., 8.9. tételek, 8.13. következmény, 8.18. tétel).*

Itt az $SR(n.i., \vee)$ osztály esetében csak diszjunkciókat alkalmazhatunk a feltételekben, míg az $SR(n.i., i., \leq k)$ osztály esetében a feltételek csupán k

	KIELÉGÍTHETŐSÉG	TARTALMAZÁS
deterministic	PTIME	PTIME
$SR()$	PTIME	coNP
$SR(n.i., \vee)$	PTIME	coNP
$SR(n.i.)$	PTIME	PSPACE-nehéz
$SR(n.i., i., \leq k)$	$\Sigma_k P$-nehéz	$\Pi_k P$-nehéz
$SR(n.i., i.)$	DEXPTIME	DEXPTIME
$SR(n.i., i., el)$	DEXPTIME	DEXPTIME

1. ábra. A kielégíthetőségi és tartalmazási kérdésre vonatkozó disszertációbeli eredmények összefoglalója.

mélységig ágyazhatók egymásba, azaz amikor a szerkezeti függvények meghívják egymást a transzformációs szabályaik feltételeiben, nem alakulhat ki kör a hívások során, ugyanakkor egy szerkezeti függvény önmagát meghívhatja a feltételeiben.

Publikációk

[1] András Benczúr and Balázs Kósa. Static analysis of structural recursions in semistructured databases and its consequences. In *ADBIS*, pages 189–203, 2004.

[2] Balás Kósa, András Benczúr, and Attila Kiss. An efficient implementation of an expressive fragment of XPath for typed and untyped XML documents using extended structural recursion. In *Local Proceedings ADBIS 2009*, pages 156–176, 2009.

[3] Balás Kósa, András Benczúr, and Attila Kiss. Satisfiability and containment problem of structural recursions with conditions. In *Proceedings of the 14th east European conference on Advances in databases and information systems*, pages 336–350. Springer-Verlag, 2010.

[4] Balázs Kósa. Containment and satisfiability problem for XPath with recursion. In *Proceedings of the 16th East European conference on Advances*

in Databases and Information Systems, pages 240–253. Springer-Verlag, 2012.

[5] Balázs Kósa, András Benczúr, and Attila Kiss. Extended structural recursion and XSLT. *Acta Univ. Sapientiae, Inform.*, 1(2):165–213, 2009.

Hivatkozások

[6] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau, and John Cowan. Extensible Markup Language (XML) 1.1 W3C Recommendation, The World Wide Web Consortium. `http://www.w3.org/TR/2004/REC-xml11-20040204/`, 2004.

[7] Val Breazu-Tannen, Peter Buneman, and Shamim Naqvi. Structural recursion as a query language. In *Proceedings of the third international workshop on Database programming languages : bulk types & persistent data: bulk types & persistent data*, DBPL3, pages 9–19. Morgan Kaufmann Publishers Inc., 1992.

[8] Peter Buneman, Susan Davidson, Mary Fernandez, and Dan Suciu. Adding structure to unstructured data. In *In 6th Int. Conf. on Database Theory (ICDT '97), LNCS 1186*, pages 336–350. Springer, 1997.

[9] Peter Buneman, Mary Fernandez, and Dan Suciu. UnQL: a query language and algebra for semistructured data based on structural recursion. *The VLDB Journal*, 9:76–110, 2000.

[10] Peter Buneman, Shamim Naqvi, Val Tannen, and Limsoon Wong. Principles of programming with complex objects and collection types. *Theor. Comput. Sci.*, 149(1):3–48, 1995.

[11] Georg Gottlob, Christoph Koch, and Reinhard Pichler. Efficient algorithms for processing XPath queries. *ACM Trans. Database Syst.*, 30(2):444–491, 2005.

[12] Neil Immerman, Sushant Patnaik, and David Stemple. The expressiveness of a family of finite set languages. *Theoretical Computer Science*, 155(1):111 – 140, 1996.

[13] Leonid Libkin and Limsoon Wong. Query languages for bags and aggregate functions. *J. Comput. Syst. Sci.*, 55(2):241–272, 1997.

[14] Wim Martens, Frank Neven, Thomas Schwentick, and Geert Jan Bex. Expressiveness and complexity of XML Schema. *ACM Trans. Database Syst.*, 31:770–813, 2006.

[15] Thomas Schwentick. XPath query containment. *SIGMOD Rec.*, 33(1):101–109, 2004.